封装方法题

1. 怎么获取地址栏url，封装成方法？

window.location.host; //返回url 的主机部分，例如：www.xxx.com

window.location.hostname; //返回www.xxx.com

window.location.href; //返回整个url字符串(在[浏览器](http://www.daimajiayuan.com/sitejs-5791-1.html" \t "http://www.360doc.com/content/16/1228/09/_blank)中就是完整的地址栏)，例如：www.xxx.com/index.php?class\_id=3&id=2

window.location.pathname; //返回/a/index.php或者/index.php

window.location.protocol; //返回url 的协议部分，例如： http:，ftp:，maito:等等。

window.location.port //url 的端口部分，如果采用默认的80端口，那么返回值并不是默认的80而是空字符

请编写一个JavaScript函数parseQueryString，它的用途是把URL参数解析为一个对象

function parseQueryString(str){

// var str=window.location.href;//获取url

var arr=str.split('?');//切割，获取参数

var str1=arr[1];

var arr1=str1.split('&');//切割，获取每一个参数

var arr2=[];

var obj={};

for(var i=0;i<arr1.length;i++){

arr2[i]=arr1[i].split('=')

}

for(var j=0;j<arr2.length;j++){

obj[arr2[j][0]]=arr2[j][1]

}

return obj;

}

1. 怎么把一组数据按日期排序？

<ul>

<li>2011-1-1</li>

<li>2011-1-7</li>

<li>2011-1-2</li>

<li>2011-3-1</li>

<li>2011-2-1</li>

</ul>

<script type='text/javascript'>

var ul = document.getElementsByTagName('ul')[0];

var lis = ul.getElementsByTagName('li');

var ux = [];

for (var i=0; i<lis.length; i++){

var tmp = {};

tmp.dom = lis[i];

tmp.date = new Date(lis[i].firstChild.data.replace(/-/g,'/'));//设置日期

ux.push(tmp);

}

ux.sort(function(a,b){//排序

return a.date - b.date;

});

for (var i=0; i<ux.length; i++){//重新渲染

ul.appendChild(ux[i].dom);

}

</script>

3.怎么获取整个表单所有的数据？

见案例

4. 怎么从excel表格中获取数据？

见案例

5.把一个函数的arguments转化成真正的数组

Array.prototype.slice.call(arguments)

6.请用js写出你知道的获取随机数的方法

Math.random()，结果为0-1之间的一个随机数（包括0，不包括1）

生成一个0-100之间的随机整数：

parseInt(Math.random()\*100)

生成一个m-n之间的随机整数：

parseInt(Math.random()\*(n-m+1)+m)

生成指定位数的随机整数：

function randomNum(n){

var t=’’;

for(var i=0;i<n;i++){

r+=Math.floor(Math.random()\*10);

}

return t;

}

生成制定范围内的随机整数：

function randomNum(minNum,maxNum){

switch(arguments.length){

case 1:

return parseInt(Math.random()\*minNum+1);

break;

case 2:

return parseInt(Math.random()\*(maxNum-minNum+1)+minNum);

break;

default:

return 0;

break;

}

}

7.javascript里面的继承怎么实现，如何避免原型链上的对象共享？

通过增加一个空的函数来作为中间量，进而避免原型链上的对象共享。

function Cat(name){

  Animal.call(this);

  this.name = name || 'Tom';

}

(function(){

  // 创建一个没有实例方法的类

  var Super = function(){};

  Super.prototype = Animal.prototype;

  //将实例作为子类的原型

  Cat.prototype = new Super();

})();

8.一个含有n个数的正整数数组（无序），要求找出两个数的和等于给定值。

方法一：排序的思想，两两加起来去和给定值进行比较

var arr=[1,2,3,4,6,7,8,9,5];

function sum(arr,num){

for(var i=0;i<arr.length;i++){

for(var j=i+1;j<arr.length;j++){

if(arr[i]+arr[j]==num){

return console.log(arr[i],arr[j])

}

}

}

}

sum(arr,14)

方法二：求两个数字之和，假定给定的和为Sum。一个变通的思路，就是对数组中的每个数字arr[i]都判别Sum-arr[i]是否在数组中，这样就变通成为一个查找的算法。可以采用先排序，后二分法的思路去查找。

9.请用es5与es6分别封装一个数组去重的方法

var arr=[1,2,1,3,5,2,5,1,7,8,5];

//es5方法

function remRep(arr){

var newArr=[];

for(var i=0,len=arr.length;i<len;i++){

if(newArr.indexOf(arr[i])==-1){

newArr.push(arr[i])

}

}

return newArr;

}

console.log(remRep(arr))//[1,2,3,5,7,8]

//es6方法

var remRep1=(arr)=>{

let newArr=[];

newArr=new Set(arr);

return newArr;

}

console.log(remRep1(arr))//[1,2,3,5,7,8]

10.封装一个对象深拷贝的方法

**浅拷贝：拷贝的是对象的属性的引用，而不是对象本身**

//**es5方法**

function simpleClone(initalObj) {

var obj = {};

for ( var i in initalObj) {

obj[i] = initalObj[i];

}

return obj;

}

console.log(simpleClone(person))//{name:'xiaoming',age:15}

//**es6方法**：

var person={name:'xiaoming',age:15}

deepCopy=(obj)=>{

let obj1={};

Object.assign(obj1,obj)//第一个为空对象时则是复制，为浅拷贝

return obj1;

}

console.log(deepCopy(person))//{name:'xiaoming',age:15}

**深拷贝：拷贝的是对象本身**

//-----深拷贝-----

// 使用 **JSON.parse()** 方法

function deepClone(initalObj) {

var obj = {};

try {

obj = JSON.parse(JSON.stringify(initalObj));

}

catch (err) {

return

}

return obj;

}

console.log(deepClone(person))//{name:'xiaoming',age:15}

/\*

这种方法简单易用。

但是这种方法也有不少坏处，譬如它会抛弃对象的constructor。也就是深拷贝之后，不管这个对象原来的构造函数是什么，在深拷贝之后都会变成Object。

这种方法能正确处理的对象只有 Number, String, Boolean, Array, 扁平对象，即那些能够被 json 直接表示的数据结构。RegExp对象是无法通过这种方式深拷贝。

\*/

//**递归拷贝**

function deepClone1(initalObj, finalObj) {

var obj = finalObj || {};

for (var i in initalObj) {

var prop = initalObj[i];

// 避免相互引用对象导致死循环，如initalObj.a = initalObj的情况

if(prop === obj) {

continue;

}

if (typeof prop === 'object') {

obj[i] = (prop.constructor === Array) ? [] : {};

arguments.callee(prop, obj[i]);

} else {

obj[i] = prop;

}

}

return obj;

}

console.log(deepClone1(person,{}))//{name:'xiaoming',age:15}

//使用**Object.create()**方法

//直接使用var newObj = Object.create(oldObj)，可以达到深拷贝的效果。

function deepClone2(initalObj, finalObj) {

var obj = finalObj || {};

for (var i in initalObj) {

var prop = initalObj[i];

// 避免相互引用对象导致死循环，如initalObj.a = initalObj的情况

if(prop === obj) {

continue;

}

if (typeof prop === 'object') {

obj[i] = (prop.constructor === Array) ? [] : Object.create(prop);

} else {

obj[i] = prop;

}

}

return obj;

}

console.log(deepClone2(person,{}))//{name:'xiaoming',age:15}

//参考：**jQuery.extend()方法**的实现,jQuery.js的jQuery.extend()也实现了对象的深拷贝

jQuery.extend = jQuery.fn.extend = function() {

var options, name, src, copy, copyIsArray, clone,

target = arguments[ 0 ] || {},

i = 1,

length = arguments.length,

deep = false;

// Handle a deep copy situation

if ( typeof target === "boolean" ) {

deep = target;

// Skip the boolean and the target

target = arguments[ i ] || {};

i++;

}

// Handle case when target is a string or something (possible in deep copy)

if ( typeof target !== "object" && !jQuery.isFunction( target ) ) {

target = {};

}

// Extend jQuery itself if only one argument is passed

if ( i === length ) {

target = this;

i--;

}

for ( ; i < length; i++ ) {

// Only deal with non-null/undefined values

if ( ( options = arguments[ i ] ) != null ) {

// Extend the base object

for ( name in options ) {

src = target[ name ];

copy = options[ name ];

// Prevent never-ending loop

if ( target === copy ) {

continue;

}

// Recurse if we're merging plain objects or arrays

if ( deep && copy && ( jQuery.isPlainObject( copy ) ||

( copyIsArray = jQuery.isArray( copy ) ) ) ) {

if ( copyIsArray ) {

copyIsArray = false;

clone = src && jQuery.isArray( src ) ? src : [];

} else {

clone = src && jQuery.isPlainObject( src ) ? src : {};

}

// Never move original objects, clone them

target[ name ] = jQuery.extend( deep, clone, copy );

// Don't bring in undefined values

} else if ( copy !== undefined ) {

target[ name ] = copy;

}

}

}

}

// Return the modified object

return target;

};

1. 下面这段代码想要循环延时输出结果0 1 2 3 4，请问输出结果是否正确，如果不正确，请说明为什么？并修改循环内的代码，使其输出正确的值。

for(var i=0;i<5;++i){

setTimeout(function(){

Console.log(i+’’)

},100)

}

输出5个5

定时器是异步加载，还没有执行定时器，for循环就已经执行完毕，所以最后在调用时候，总是调用for循环的最后一个值

for(var i=0;i<5;++i){

(function(j){

setTimeout(function(){

console.log(j+'')//0,1,2,3,4

},100)

})(i)

}

12.实现一个函数，返回今天星期几（例如今天周一，返回字符串“今天是星期一”）

function getWeek(){

var weekDay=new Date().getDay()

switch(weekDay){

case 0:

console.log('今天是星期日')

break;

case 1:

console.log('今天是星期一')

break

case 2:

console.log('今天是星期二')

break

case 3:

console.log('今天是星期三')

break

case 4:

console.log('今天是星期四')

break

case 5:

console.log('今天是星期五')

break

case 6:

console.log('今天是星期六')

break

}

}

13.建立一个数组，生成100个保留2位小数的随机数添加到这个数组中，然后对这个数组去重，然后从大到小排序。

var arr=[];

var arr1=[];

for(var i=0;i<100;i++){

arr.push(Math.random().toFixed(2));

}

for(var j=0;j<arr.length;j++){

if(arr1.indexOf(arr[j])==-1){

arr1.push(arr[j])

}

}

arr1.sort(function(a,b){

return b-a;

})

14.实现一个函数，传入一个字符串使这个字符串倒序排列并返回。

function reverse(str){

var newStr=str.split('').reverse().join('');

return newStr;

}